Link

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-26

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 5454 bytes

Attack Category	• Privilege Exploitation
Vulnerability Category	• Indeterminate File/Path • TOCTOU - Time of Check, Time of Use
Software Context	• File Management
Location	• unistd.h
Description	The link(name1, name2) call atomically creates the specified directory entry (hard link) name2 with the attributes of the underlying object pointed at by name1. Because link() references the underlying filesystem object to be linked to by name, it is vulnerable to an attacker substituting an unexpected file to be linked to.
APIs	**Function Name** | **Comments** link |
Method of Attack	The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results. If a setuid-root program attempts to link to a file that an attacker can replace with a link to another file, the attacker could trick the program into acting on an file that the attacker would not otherwise have access to.
Exception Criteria	
Solutions	**Solution Applicability** | **Solution Description** | **Solution Efficacy**

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

Whenever a privileged program invokes link().	Ensure that (1) the file to be linked to cannot be tampered with, or (2) reduce privileges to ensure that linking will not provide inappropriate access, or (3) perform checks after linking to verify that the correct file was linked to.	Likely to be effective.
Generally applicable.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.
Generally applicable.	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
Generally applicable.	Limit the spread of time (cycles) between the check and use of a resource.	Does not eliminate the underlying vulnerability but can help

		make it more difficult to exploit.
Generally applicable.	Recheck the resource after the use call to verify that the action was taken appropriately.	Checking the filename permissions after the operation does not change the fact that the operation may have been exploited but it does allow halting of the application in an error state to help limit further damage.

Signature Details	int link(const char *name1, const char *name2);

Examples of Incorrect Code	`// assume setuid-root`
	`link(name1, name2);`
	`// operate on name1`

Examples of Corrected Code	`// assume setuid-root`
	`// reduce privileges`
	`link(name1, name2);`
	`// operate on name1`

Source References	• ITS4 Source Code Vulnerability Scanning Tool[2] • http://seclab.cs.ucdavis.edu/projects/vulnerabilities/scriv/ucd-ecs-95-09.pdf[3]
Recommended Resource	man page for link()[4]
Discriminant Set	**Operating System**: • Any **Languages**: • C • C++

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com